

# CrimeBoard (Use Cases, SQL Query & Relational Algebra Expression)

**Class:** DAMG6210

**CRN:** 16149

**Team members:**

Sayed Ahmed - 002191535

Natarajan Lekshmi Narayana Pillai - 002766033

Chaman Betrabet – 002784662

Sunil Kumar Rudrakumar - 002764807

**1)**

**Use Case:** How many tweets did the users make in the past 24 hours?

**Description:** Use the tweet\_userhist table to show the number of tweets made in 24hrs by a user.

**Actor:** User

**Actor action:** The user views the number of tweets made in 24hrs by a user.

**System Responses:** the number of tweets made in 24hrs by a user is displayed

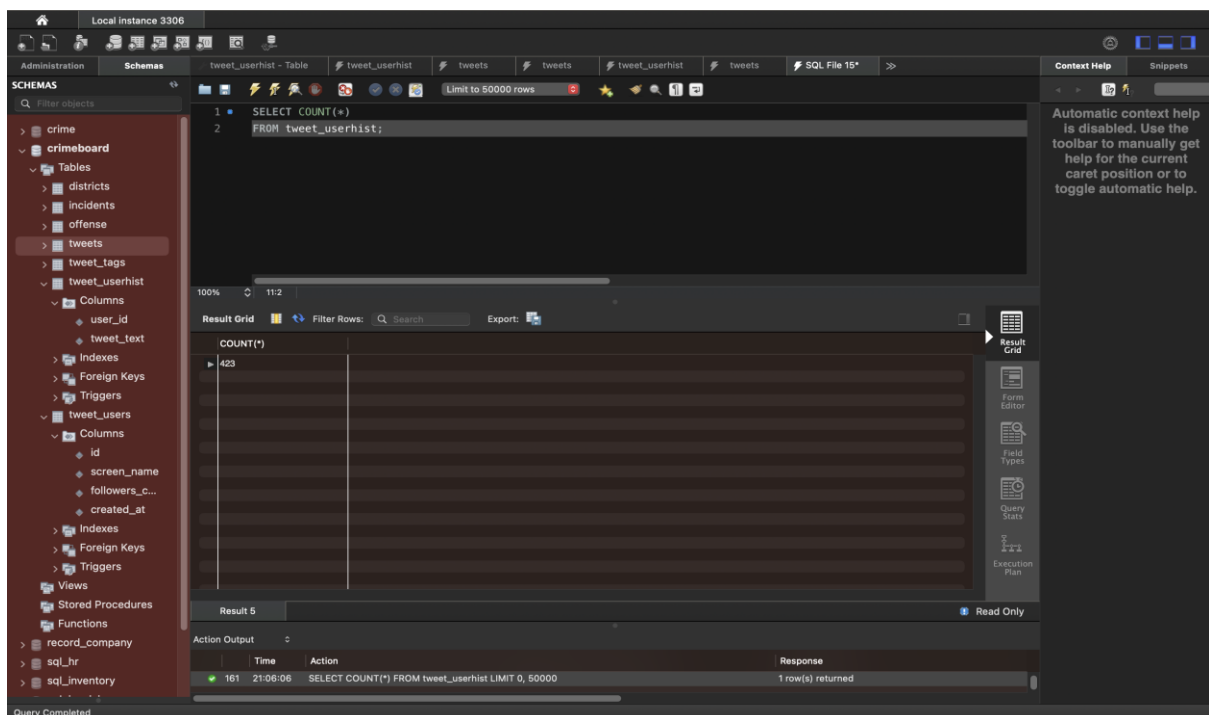
**Post Condition:** system displays the number of tweets made in 24hrs by a user.

## SQL Query

*SELECT COUNT(\*) FROM tweet\_userhist;*

## Relational Algebra Expression

$\pi_{tweet\_count} tweet\_userhist$



The screenshot displays the DBeaver database client interface. At the top, the title bar indicates the local instance is 'Local instance 3306'. The main window is divided into several panes:

- Left Pane (Schemas):** A tree view showing the database structure. The 'crimeboard' schema is selected, and the 'tweets' table is highlighted under the 'Columns' section.
- Top Center Pane (SQL Editor):** Contains a single SQL query: `SELECT favorite_count FROM crimeboard.tweets;`. The query is limited to 50,000 rows.
- Bottom Center Pane (Result Grid):** Displays the query results in a table format. The table has one column, 'favorite\_count', and 16 rows of data. The first few values are 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 35, 0, 2, 1, 0, 0.
- Right Pane (Context Help):** A sidebar with a message: 'Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.'
- Bottom Right Pane (Action Output):** Shows the execution details of the query. It indicates that 162 rows were returned in 21:07:18 seconds.

The interface is dark-themed, and the overall layout is typical of a database management tool.

3)

**Use Case:** What are the retweet counts of tweets?

**Description:** Use the tweets table to show the number of retweets of tweets.

**Actor:** User

**Actor action:** The user views the number of retweets of tweets.

**System Responses:** The number of retweets of tweets is displayed

**Post Condition:** system displays the number of retweets of tweets.

## SQL Query

*SELECT retweet\_count FROM tweets;*

## Relational Algebra Expression

$\pi_{retweet\_count} tweets$

The screenshot shows a database management tool interface. On the left, a 'SCHEMAS' panel lists various database objects, including tables like 'tweets' and 'tweet\_userhist'. The main area displays a SQL query: `SELECT retweet_count FROM crimeboard.tweets;`. Below the query, a 'Result Grid' shows the output of the query, which is a single column labeled 'retweet\_cou...' with 705 rows of data. The bottom of the interface shows an 'Action Output' panel with a log entry: '163 21:08:33 SELECT retweet\_count FROM crimeboard.tweets LIMIT 0, 50000 705 row(s) returned'. The status bar at the bottom indicates 'Query Completed'.

4)

**Use Case:** When was the latest tweet created?

**Description:** Use the tweets table to show the latest tweet created.

**Actor:** User

**Actor action:** The user views the latest tweet created.

**System Responses:** The latest tweet created is displayed

**Post Condition:** system displays the latest tweet created.

## SQL Query

```
SELECT id_str, text, id, max(created_at)
```

```
FROM tweets
```

```
GROUP BY id_str, text, id, created_at
```

## Relational Algebra Expression

$\pi_{id\_str, text, id, MAX(created\_at)}$

$\gamma_{id\_str, text, id, created\_at, MAX(created\_at)} tweets$

The screenshot shows a database management tool interface with a dark theme. On the left, a 'SCHEMAS' sidebar lists various database objects, including 'crimeboard', 'tweets', and 'tweet\_userhist'. The main window displays a SQL query in a text editor:

```
1 select id_str, text, id, max(created_at)
2 from crimeboard.tweets
3 group by id_str, text, id, created_at;
```

Below the query editor, a 'Result Grid' shows the results of the query. The grid has four columns: 'id\_str', 'text', 'id', and 'max(created\_at)'. It contains multiple rows of data, including tweets about Pakistan PM Imran Khan, a gunshot, and actor Jon-Erik Hexum. The bottom of the interface shows a status bar indicating 'Query Completed' and '705 row(s) returned'.

5)

**Use Case:** Does a single user tweet multiple times?

**Description:** Use the tweets table to show does a single user tweets multiple times.

**Actor:** User

**Actor action:** The user views a single user's tweet multiple times.

**System Responses:** does a single user tweet multiple times is displayed

**Post Condition:** system displays does a single user tweets multiple time

## SQL Query

```
SELECT id_str, text, COUNT(*)id
FROM tweets
GROUP BY id_str
```

## Relational Algebra Expression

$\pi$  id\_str, text, id

$\gamma$  id\_str, tweets

The screenshot shows a database management tool interface. The top bar indicates 'Local Instance 3306'. The left sidebar shows a 'SCHEMAS' tree with folders for 'crime', 'crimeboard', 'Tables', 'districts', 'incidents', 'offense', 'tweets', 'tweet\_tags', 'tweet\_userhist', 'Columns', 'Indexes', 'Foreign Keys', 'Triggers', 'tweet\_users', 'Columns', 'id', 'screen\_name', 'followers\_c...', 'created\_at', 'Indexes', 'Foreign Keys', 'Triggers', 'Views', 'Stored Procedures', 'Functions', 'record\_company', 'sql\_hr', and 'sql\_inventory'. The main area displays a SQL query: 

```
1 SELECT id_str, text, COUNT(*)id
2 FROM tweets
3 GROUP BY id_str;
```

 Below the query is a 'Result Grid' showing a table with columns 'id\_str', 'text', and 'id'. The table contains 705 rows of data, including tweets from users like 'Pakistan PM Imran Khan' and 'DrBuz'. The bottom status bar shows 'Query Completed' and '705 row(s) returned'.

6)

**Use Case:** How many users tweeted about crime?

**Description:** Use the tweet\_users table to show how many users tweeted about crime.

**Actor:** User

**Actor action:** The user views how many users tweeted about crime.

**System Responses:** How many users tweeted about the crime is displayed.

**Post Condition:** system displays How many users tweeted about crime.

## SQL Query

*SELECT COUNT(\*) FROM tweet\_users*

## Relational Algebra Expression

$\pi$  COUNT(\*)

$\gamma$  COUNT(\*) tweet\_users

The screenshot shows a database management interface with a dark theme. On the left, a 'SCHEMAS' sidebar lists various database objects, including 'crimeboard', 'tweet\_tags', 'tweet\_userhist', and 'tweet\_users'. The main area displays a SQL query: `select count(*) from tweet_users;`. Below the query, a 'Result Grid' shows a single row with the value '552'. At the bottom, an 'Action Output' section shows the query execution details: '166 21:13:21 select count(\*) from tweet\_users LIMIT 0, 50000' and '1 row(s) returned'. A 'Query Completed' status is visible at the very bottom.

7)

**Use Case:** How many tweets have the keyword crime?

**Description:** Use the tweets table to show how many tweets have the keyword crime.

**Actor:** User

**Actor action:** The user views how many tweets have the keyword crime

**System Responses:** How many tweets have the keyword crime are displayed.

**Post Condition:** system displays how many tweets that have the keyword crime

## SQL Query

```
SELECT COUNT(*) FROM tweets WHERE TEXT LIKE '%crime%'
```

## Relational Algebra Expression

$\pi$  COUNT (\*)

$\gamma$  COUNT (\*)

$\sigma$  text LIKE "%crime%" tweets

The screenshot shows a database management tool interface. On the left, a 'SCHEMAS' panel displays a tree view of the database structure, including tables like 'crimeboard', 'districts', 'incidents', 'offense', 'tweets', 'tweet\_tags', 'tweet\_userhist', 'tweet\_users', 'record\_company', 'sql\_hr', and 'sql\_inventory'. The 'tweets' table is selected. The main query editor displays the SQL query: `select count(*) from tweets where text like 'crime%';`. Below the query editor, the 'Result Grid' shows a single row with the value '93' under the column 'count(\*)'. At the bottom, the 'Action Output' panel shows the execution details: '167 21:14:01 select count(\*) from tweets where text like 'crime%'; LIMIT 0, 50000' and '1 row(s) returned'. A 'Context Help' panel on the right provides additional information.

count(*)
93

Time	Action	Response
167 21:14:01	select count(*) from tweets where text like 'crime%'; LIMIT 0, 50000	1 row(s) returned

8)

**Use Case:** Is there a police department official handle tweeting in our records?

**Description:** Use the tweet\_users table to show whether Is there a police department official handling tweeting in our records.

**Actor:** User

**Actor action:** The user views whether there is a police department official handle tweeting in our records.

**System Responses:** Is there a police department official handle tweeting in our records displayed.

**Post Condition:** system displays whether there is a police department official handle tweeting in our records

### SQL Query

```
SELECT * FROM tweet_users WHERE screen_name LIKE '%pd'
```

### Relational Algebra Expression

$\pi$  COUNT (\*)

$\gamma$  COUNT (\*)

$\sigma$  screen\_name LIKE "%PD%" tweet\_users

The screenshot shows a database management interface with a dark theme. On the left, a 'SCHEMAS' sidebar lists a hierarchy of databases: 'crime', 'crimeboard', 'districts', 'incidents', 'offense', 'tweets', 'tweet\_tags', 'tweet\_userhist', 'tweet\_users', 'tweet\_text', 'Indexes', 'Foreign Keys', 'Triggers', 'Views', 'Stored Procedures', 'Functions', 'record\_company', 'sql\_hr', and 'sql\_inventory'. The 'tweet\_users' table is selected. The main area displays a SQL query: `SELECT * FROM crimeboard.tweet_users WHERE screen_name LIKE '%pd';`. Below the query, a 'Result Grid' shows the query results. The grid has four columns: 'id', 'screen\_name', 'followers\_cou...', and 'created\_at'. The first row shows values: '171', 'HALL', 'HALL', and 'HALL'. The bottom status bar indicates 'Query Completed' and '0 row(s) returned'.



9)

**Use Case:** Which city had the most tweets on crime?

**Description:** Use the tweets table to show which city had the most tweets on crime.

**Actor:** User

**Actor action:** The user views which city had the most tweets on crime

**System Responses:** which city had the most tweets on crime are displayed.

**Post Condition:** system displays which city had the most tweets on crime

## SQL Query

```
SELECT city, count(id_str) FROM tweets  
GROUP BY city;
```

## Relational Algebra Expression

$\gamma_{city, COUNT(id\_str) tweets}$

The screenshot shows a database management tool interface. On the left, a 'SCHEMAS' panel lists various database objects, including 'crimeboard' and 'tweets'. The main area displays a SQL query: `SELECT city, count(id_str) from crimeboard.tweets group by city;`. Below the query, a 'Result Grid' shows the results of the query, with columns 'city' and 'count(id\_str)'. The results are as follows:

city	count(id_str)
13 St Mary Ave	238
United States	32
Office No: 401 Bay Street	1
Schaumburg	1
Brooklyn	1
Manchester	1
Lucknow	1
Bradenton	1
Teynampet	1
Gilligan's Island via satellite	1
Nuevo Leon	1
one admin / a)	1
Seattle	2
Chennai	1
Pattaling Jaya	1
Andhra Pradesh	5
Vijayawada	2
Kakinada	1
India	1
kadapa	2
Tirupati	1
Bengaluru	3
Cuddapah	2
Visakhapatnam	2
Iran Abadan	14

At the bottom, an 'Action Output' panel shows the query execution details: '172 21:16:24 SELECT city, count(id\_str) from crimeboard.tweets group by city LIMIT 0, 50000' and '281 row(s) returned'.

10)

**Use Case:** How many tweets were tweeted on the current date?

**Description:** Use the tweets table to show how many tweets were tweeted on the current date

**Actor:** User

**Actor action:** The user views how many tweets were tweeted on the current date.

**System Responses:** how many tweets were tweeted on the current date is displayed.

**Post Condition:** system displays how many tweets were tweeted on the current date.

## SQL Query

*SELECT \* FROM tweets WHERE date(created\_at) = current\_date();*

## Relational Algebra Expression

$\sigma_{\text{created\_at} = \text{current\_date}} \text{ tweets}$

The screenshot displays a database management interface with a dark theme. On the left, a 'SCHEMAS' sidebar shows a tree view of the database structure, including 'crimeboard' and 'tweets'. The main window shows a SQL query: `SELECT * FROM crimeboard.tweets WHERE date(created_at) = current_date();`. Below the query, a 'Result Grid' displays a list of tweets with columns for 'id\_str', 'text', 'created\_at', 'retweet\_count', and 'favorite\_count'. The 'text' column contains various tweets, some with links and hashtags. The 'created\_at' column shows timestamps. The 'retweet\_count' and 'favorite\_count' columns show numerical values. On the right, a sidebar contains 'Context Help' and 'Snippets' tabs. At the bottom, an 'Action Output' section shows the execution of the query, indicating that 369 rows were returned.

11)

**Use Case:** How many tweets were tweeted at the current hour?

**Description:** Use the tweets table to show how many tweets were tweeted on the current date

**Actor:** User

**Actor action:** The user views how many tweets were tweeted on the current date.

**System Responses:** how many tweets were tweeted on the current date is displayed.

**Post Condition:** system displays how many tweets were tweeted on the current date.

## SQL Query

```
SELECT * FROM tweets WHERE hour(created_at) = hour(current_time());
```

## Relational Algebra Expression

$\sigma_{\text{created\_at} = \text{hour tweets}}$

The screenshot shows a database management interface with a sidebar on the left containing a schema tree. The main area displays a SQL query: `SELECT * FROM crimeboard.tweets WHERE hour(created_at) = hour(current_time());`. Below the query, a 'Result Grid' shows a table of tweet data with columns: `id_str`, `text`, `created_at`, `retweet_count`, and `favorite_count`. The table contains 15 rows of data, including tweets about the Paris Conference of Hugo Salgari, a school fight, and a reminder to check out a guide for law enforcement executives. The bottom of the interface shows an 'Action Output' section with a green status icon, a timestamp of 21:18:31, and a message: 'SELECT \* FROM crimeboard.tweets WHERE hour(created\_at) = hour(current\_time()) LIMIT 0, 50000 48 row(s) returned'.

12)

**Description:** What date had the most tweets about crime?

**Actor:** User

**Actor action:** The user views what date had the most tweets about crime

**System Responses:** what date had the most tweets about crime are displayed.

**Post Condition:** system displays what date had the most tweets about crime

### SQL Query

```
SELECT created_at FROM tweets GROUP BY created_at  
ORDER BY created_at DESC;
```

### Relational Algebra Expression

$\tau_{created\_at} \downarrow$

$\gamma_{created\_at, COUNT(created\_at)} tweets$

The screenshot shows a database management interface with a dark theme. On the left, a 'SCHEMAS' sidebar lists a hierarchy: 'crime' > 'crimeboard' > 'Tables' > 'tweets'. The main window displays a SQL query in a text editor: 

```
1 SELECT created_at FROM crimeboard.tweets GROUP BY created_at  
2 ORDER BY created_at DESC;
```

 Below the editor is a 'Result Grid' showing a list of timestamps from '2022-11-13 01:45:45' down to '2022-11-13 01:38:17'. At the bottom, an 'Action Output' pane shows a log entry: '177 21:21:28 SELECT created\_at FROM crimeboard.tweets GROUP BY created\_at ORDER BY created\_at DESC LIMIT... 695 row(s) returned'. A status bar at the very bottom indicates 'Query Completed'.

13)

**Use Case:** Which city had the most tweets on crime?

**Description:** Use the tweets table to show which city had the most tweets on crime.

**Actor:** User

**Actor action:** The user views which city had the most tweets on crime.

**System Responses:** which city had the most tweets on crime is displayed.

**Post Condition:** system displays which city had the most tweets on crime.

## SQL Query

*SELECT city, count(city) from tweets group by city  
order by city DESC;*

## Relational Algebra Expression

$\tau_{city} \downarrow$

$\gamma_{city, COUNT(city) tweets}$

The screenshot shows a database management tool interface. On the left, a 'SCHEMAS' panel lists various database objects, including 'crime', 'crimeboard', 'districts', 'incidents', 'offense', 'tweets', 'tweet\_tags', 'tweet\_userhist', 'Columns', 'Indexes', 'Foreign Keys', 'Triggers', 'tweet\_users', 'Columns', 'Indexes', 'Foreign Keys', 'Triggers', 'Views', 'Stored Procedures', 'Functions', 'record\_company', 'sql\_hr', and 'sql\_inventory'. The main area displays a SQL query: `1 SELECT city, count(city) from tweets group by city` and `2 order by city DESC;`. Below the query, a 'Result Grid' shows the results of the query, with columns 'city' and 'count(cit...'. The results are sorted by city in descending order. The bottom panel shows the 'Action Output' with a table containing 'Time' and 'Response' columns. The response indicates that 281 row(s) were returned.

city	count(cit...
大連だったらしいな	1
サクセスセブンブルク三丁目	1
Укрaina	1
Москва	1
в недалеке на диком диком	1
Ayivo	1
Yaman	1
YB GUTS	1
Wrapped around Jiwoo's finger	1
Worldwide	5
Winchester	1
We're In A Race For Relevancy	1
Washington dc	1
Washington	1
Visit my book blog RLFblog.com	1
Visakhapatnam	2
Virginia	10
Vijayawada	2
Vermont	1
Vancouver	4
UT 10 385907	1
USA	7
US New York	1
US	1
United States	1
United States	32

14)

**Use Case:** Which state had the most tweets on crime?

**Description:** Use the tweets table to show which state had the most tweets on crime.

**Actor:** User

**Actor action:** The user views which state had the most tweets on crime.

**System Responses:** which state had the most tweets on crime is displayed.

**Post Condition:** system displays which state had the most tweets on crime.

## SQL Query

*SELECT state, count(state) from tweets group by state  
order by state DESC;*

## Relational Algebra Expression

$\tau_{state} \downarrow$

$\gamma_{state, COUNT(state)} tweets$

The screenshot shows a database management tool interface. On the left, a 'SCHEMAS' panel lists various tables including 'crime', 'crimeboard', 'districts', 'incidents', 'offense', 'tweets', 'tweet\_tags', 'tweet\_userhist', 'tweet\_users', 'record\_company', 'sql\_hr', and 'sql\_inventory'. The 'tweets' table is selected. The main area displays a SQL query: `1 SELECT state, count(state) from tweets group by state` and `2 order by state DESC;`. Below the query, a 'Result Grid' shows the results of the query, with columns 'state' and 'count(stat...'. The results are sorted by state in descending order. The bottom status bar indicates 'Query Completed'.

state	count(stat...
Arb - Saudi Arabia	1
99-249416	1
IBB	2
Wynberg	2
WA/Scottsdale	1
WA	5
VT	1
VA	3
USA	22
United Kingdom	3
United Arab Emir...	1
UK	2
TX	9
Thailand	1
Texas	2
Sverige	1
Swiss	1
South Africa	1
Slovakia	14
SC	3
Patagonia	1
Pakistan	2
PA	1
OR	1
Ontario	1
Chis	1

15)

**Use Case:** Which country had the most tweets on crime?

**Description:** Use the tweets table to show which country had the most tweets on crime.

**Actor:** User

**Actor action:** The user views which country had the most tweets on crime.

**System Responses:** which country had the most tweets on crime is displayed.

**Post Condition:** system displays which country had the most tweets on crime.

## SQL Query

*SELECT country, count(country) from tweets group by country  
order by country DESC;*

## Relational Algebra Expression

$\tau_{country} \downarrow$

$\gamma_{country, COUNT(country)} tweets$

The screenshot shows a database management tool interface. On the left, a 'SCHEMAS' panel lists various database objects, including 'crime', 'crimeboard', 'districts', 'incidents', 'offense', 'tweets', 'tweet\_tags', 'tweet\_userhist', 'tweet\_users', 'record\_company', 'sql\_hr', and 'sql\_inventory'. The 'tweets' table is selected. The main area displays a SQL query: `SELECT country, count(country) from tweets group by country order by country DESC;`. Below the query, a 'Result Grid' shows the results of the query. The first row is 'United States' with a count of 705. The bottom of the interface shows an 'Action Output' panel with a green checkmark, indicating the query was executed successfully. The 'Response' column shows '1 row(s) returned'.

country	count(count...)
United States	705



16)

**Use Case:** How many tweets about crime were sent between the hours of 6 pm and 6 am?

**Description:** Use the tweets table to show how many tweets about crime were sent between the hours of 6 pm and 6 am.

**Actor:** User

**Actor action:** The user views how many tweets about crime were sent between the hours of 6 pm and 6 am.

**System Responses:** how many tweets about crime were sent between the hours 6 pm and 6 am is displayed.

**Post Condition:** system displays how many tweets about crime were sent between the hours of 6 pm and 6 am.

## SQL Query

*SELECT \* FROM tweets*

*WHERE created\_at >= '2022-11-12 18:00:00.000' AND created\_at <= '2022-11-13 06:00:00.000';*

## Relational Algebra Expression

$\sigma_{created\_at \geq "2022-11-12\ 18:00:00.000" \wedge created\_at \leq "2022-11-13\ 06:00:00.000"} tweets$

The screenshot shows a database management tool interface. On the left, a sidebar displays a schema tree with folders for 'crime', 'crimeboard', 'districts', 'incidents', 'offense', 'tweets', 'tweet\_tags', 'tweet\_userhist', 'Columns', 'Indexes', 'Foreign Keys', 'Triggers', 'tweet\_users', 'Columns', 'id', 'screen\_name', 'followers\_c...', 'created\_at', 'Indexes', 'Foreign Keys', 'Triggers', 'Views', 'Stored Procedures', 'Functions', 'record\_company', 'sql\_hr', and 'sql\_inventory'. The main area shows a SQL query: `SELECT * FROM tweets WHERE created_at >= '2022-11-12 18:00:00.000' AND created_at <= '2022-11-13 06:00:00.000';`. Below the query, a 'Result Grid' displays a table with columns: 'id\_str', 'text', 'created\_at', 'retweet\_cou...', and 'favorite\_cou...'. The table contains 22 rows of tweet data. At the bottom, an 'Action Output' section shows the query execution details: '182 21:24:59 SELECT \* FROM tweets WHERE created\_at >= '2022-11-12 18:00:00.000' AND created\_at <= '2022-11-13 06:00:00.000'; 577 row(s) returned'.



17)

**Use Case:** How many tweets about crime were sent between the hours of 6 am and 6 pm?

**Description:** Use the tweets table to show how many tweets about crime were sent between the hours of 6 am and 6 pm.

**Actor:** User

**Actor action:** The user views how many tweets about crime were sent between the hours of 6 am and 6 pm.

**System Responses:** how many tweets about crime were sent between the hours 6 am and 6 pm is displayed.

**Post Condition:** system displays how many tweets about crime were sent between the hours 6 am and 6 pm.

## SQL Query

*SELECT \* FROM tweets*

*WHERE created\_at >= '2022-11-12 06:00:00.000' AND created\_at <= '2022-11-12 18:00:00.000'*

## Relational Algebra Expression

$\sigma_{created\_at \geq "2022-11-12\ 06:00:00.000" \wedge created\_at \leq "2022-11-12\ 18:00:00.000"} tweets$

The screenshot shows a database management tool interface. On the left, a sidebar displays a schema tree with a 'crime' database containing a 'tweets' table. The main area shows a SQL query: `SELECT * FROM tweets WHERE created_at >= '2022-11-12 06:00:00.000' AND created_at <= '2022-11-12 18:00:00.000'`. Below the query, a 'Result Grid' displays 21 rows of tweet data, including columns for tweet ID, text, creation time, retweet count, and favorite count. The bottom status bar indicates 'Query Completed' and '23 row(s) returned'.

18)

**Use Case:** What tweets has this user posted in the past 24 hours?

**Description:** Use the tweet\_userhist table to show what tweets have this user posted in the past 24 hours

**Actor:** User

**Actor action:** The user views what tweets have this user posted in the past 24 hours

**System Responses:** what tweets have this user posted in the past 24 hours is displayed.

**Post Condition:** system displays what tweets have this user posted in the past 24 hours

### SQL Query

*SELECT \* FROM tweet\_userhist WHERE user\_id = '1572665185319403525';*

### Relational Algebra Expression

$\sigma_{user\_id = 1572665185319403525} tweet\_userhist$

The screenshot shows a database management tool interface. On the left, a 'SCHEMAS' panel lists various database objects, including 'crimeboard', 'districts', 'incidents', 'offense', 'tweets', 'tweet\_tags', 'tweet\_userhist', 'tweet\_users', and 'views'. The 'tweet\_userhist' table is selected. The main area displays a SQL query: `select * from tweet_userhist where user_id = 1591246553771311104;`. Below the query, a 'Result Grid' shows the results of the query. The grid has two columns: 'user\_id' and 'tweet\_text'. The first row shows the user\_id '1591246553771311104' and the tweet\_text '@KBR0345 https://t.co/NpQ1C3M4/Z'. The bottom of the interface shows an 'Action Output' panel with a table containing the following data:

	Time	Action	Response
186	21:26:11	select * from tweet_userhist where user_id = 1591246553771311104 LIMIT 0, 50000	1 row(s) returned

19)

**Use Case:** How many tweets has this user posted in the past 24 hours?

**Description:** Use the tweet\_userhist table to show how many tweets have this user posted in the past 24 hours

**Actor:** User

**Actor action:** The user views how many tweets have this user posted in the past 24 hours

**System Responses:** how many tweets has this user posted in the past 24 hours is displayed.

**Post Condition:** system displays how many tweets have this user posted in the past 24 hours

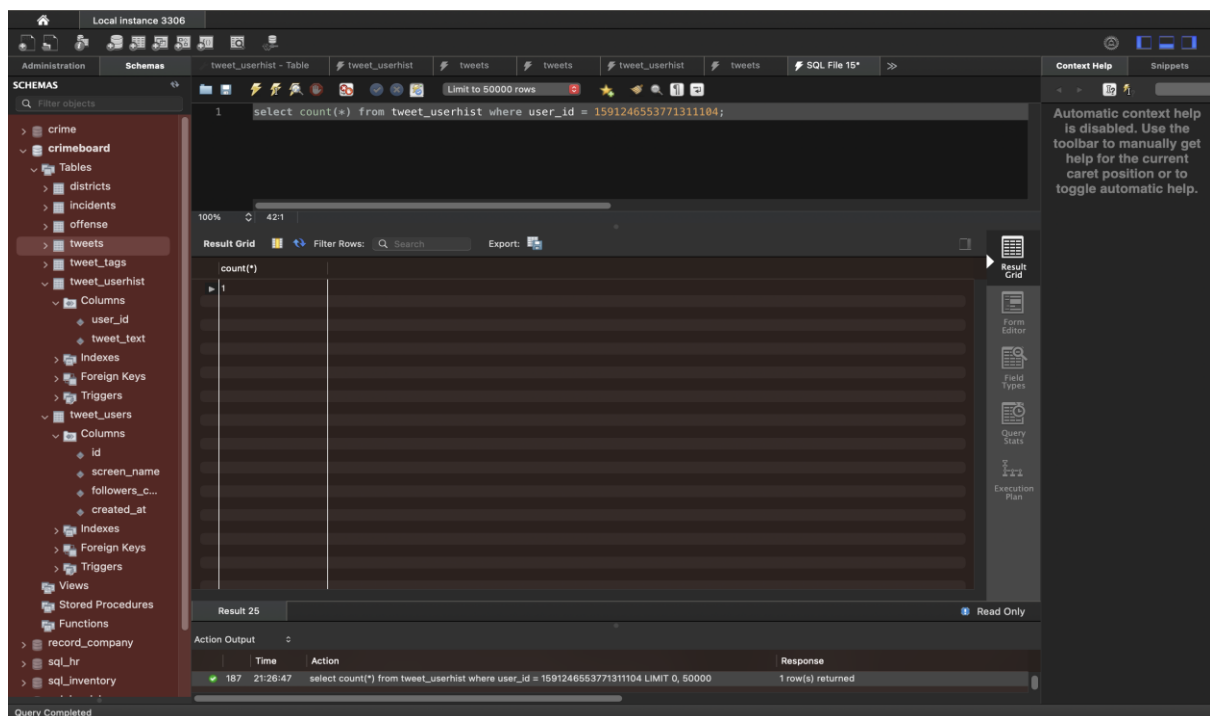
### SQL Query

*SELECT COUNT(\*) FROM tweet\_userhist WHERE user\_id = 1572665185319403525;*

### Relational Algebra Expression

$\pi$  COUNT(\*)

$\gamma$  COUNT(\*)



20)

**Use Case:** What user posted this tweet about crime?

**Description:** Use the tweets table to show what user posted this tweet about crime.

**Actor:** User

**Actor action:** The user views what the user posted in this tweet about crime.

**System Responses:** What user posted this tweet about the crime is displayed.

**Post Condition:** system displays what user posted this tweet about crime.

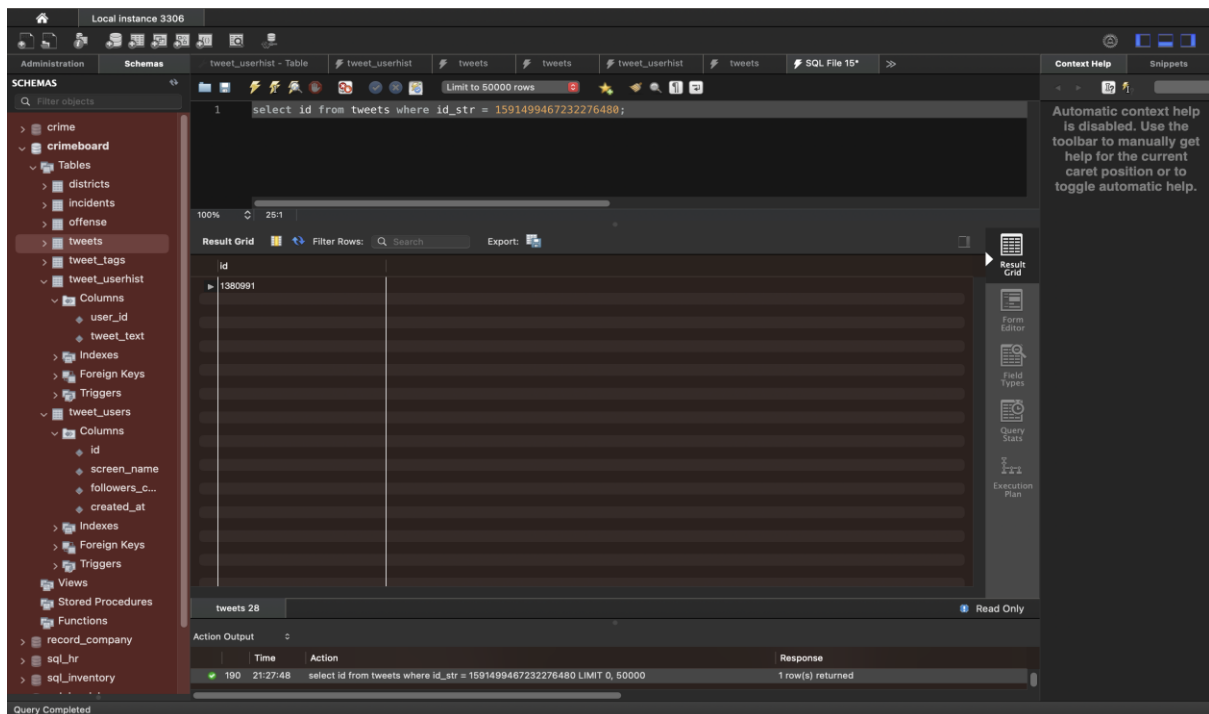
### SQL Query

*SELECT id FROM tweets WHERE id\_str = 1590902258098900992;*

### Relational Algebra Expression

$\pi$  id

$\sigma$  id\_str = 1590902258098900992 tweets



21)

**Use Case:** When did the user post this tweet about crime?

**Description:** Use the tweets table to show when did the user post this tweet about crime.

**Actor:** User

**Actor action:** The user views when the user posts this tweet about crime.

**System Responses:** when did the user post this tweet about the crime is displayed.

**Post Condition:** system displays when did the user post this tweet about crime.

## SQL Query

*SELECT created\_at FROM tweets WHERE id\_str = 1590902258098900992;*

## Relational Algebra Expression

$\pi$  created\_at

$\sigma$  id\_str = 1590902258098900992 tweets

The screenshot shows a database management tool interface. On the left, a 'SCHEMAS' panel lists various database objects, including 'crimeboard', 'crimeboard', 'districts', 'incidents', 'offense', 'tweets', 'tweet\_tags', 'tweet\_userhist', 'Columns', 'Indexes', 'Foreign Keys', 'Triggers', 'tweet\_users', 'Columns', 'id', 'screen\_name', 'followers\_c...', 'created\_at', 'Indexes', 'Foreign Keys', 'Triggers', 'Views', 'Stored Procedures', 'Functions', 'record\_company', 'sql\_hr', and 'sql\_inventory'. The main area displays a SQL query: `select created_at from tweets where id_str = 1591499467232276480;`. Below the query, a 'Result Grid' shows a single row with the value '2022-11-12 18:33:36'. The bottom panel shows the 'Action Output' with a green checkmark, indicating the query was executed successfully. The response shows '1 row(s) returned'.

created_at
2022-11-12 18:33:36

Time	Action	Response
191 21:28:34	select created_at from tweets where id_str = 1591499467232276480 LIMIT 0, 50000	1 row(s) returned